

Fechas y Tiempos

Programación 2

Trabajo con Fechas



LocalDate

- representa una fecha sin tener en cuenta el tiempo.
- Se puede crear un objeto localdate con el método:

`of(int year, int month, int dayOfMonth)`

```
LocalDate date = LocalDate.of(1979, 1, 19); //1979-01-19
System.out.println(date.getYear()); //1979
System.out.println(date.getMonth()); //JANUARY
System.out.println(date.getDayOfMonth()); //19
```

LocalDate

- Obtener la fecha Actual

```
LocalDate ld = LocalDate.now();
```

```
LocalDate ld2 = LocalDate.now();
```

Metodos Utiles

- `ld.isAfter(ld2);` //ld esta despues que ld2
- `ld.isBefore(ld2);` //ld esta despues que ld2

Trabajo con Fechas

Para dar legibilidad al código, se puede usar **Month**.



```
LocalDate date = LocalDate.of(1979, Month.JANUARY, 19);
```

Para obtener el `LocalDate` actual se usa el método `now()`:

```
LocalDate hoy = LocalDate.now();
```

Trabajo con Fechas

Y en otro idioma?

viernes, 01 enero de 2016

```
LocalDate localDate = LocalDate.of(2016,01,01);
Locale spanishLocale = new Locale("es", "ES");
String dateInSpanish=localDate.format(DateTimeFormatter.ofPattern("EEEE, dd MMMM 'de'
                                                                    yyyy",spanishLocale));
System.out.println("'2016-01-01' en Español: "+dateInSpanish);
```

Formatos

— — —

- La cantidad de letras del patron determina el formato
 - Texto
 - Menos de 4 letras usa format corto (p.ej. 'vie.')
 - Exactamente 4 letras usa format complete (p.ej. 'viernes')
 - 5 letras usa formato reducido (p.ej. 'v')
 - Número:
 - 1 letra, usa la minima cantidad de dígitos (p.ej. '5')
 - Más de una letra complete con 0 (p.ej. '05')
 - Número / Texto:
 - Si se usan 3 o más letras se aplican las reglas de texto
 - En otro caso se aplican las reglas de número

Símbolo	Significado	Presentación	Ejemplos
G	Era	Texto	AD; Anno Domini; A
u	Año	Año	2019; 19
y	Año de la era	Año	2019; 19
D	Día del año	Número	189
M / L	Mes del año	Número / Texto	7; 07; Jul; July; J
d	Día del mes	Número	10
Q / q	Trimestre del año	Número / Texto	3; 03; Q3; 3rd quarter
W	Semana del mes	Número	4
E	Día de la semana	Texto	Tue; Tuesday; T
e / c	Día de la semana localizado	Número / Texto	Viernes; 5; V
F	Semana del mes	Número	4

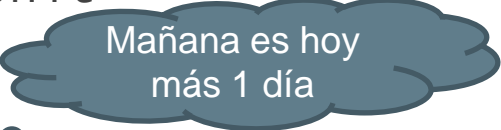
Period

La clase **Period** representa una diferencia entre dos fechas

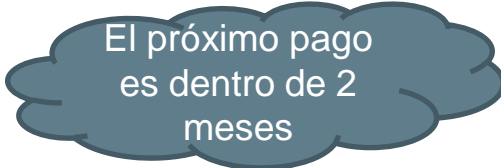
```
LocalDate hoy = LocalDate.now();  
Period dias4 = Period.ofDays(4);  
System.out.println("Hoy más 4 días es: "+hoy.plus(dias4));
```

También se puede sumar un período con ChronoUnit

```
LocalDate hoy = LocalDate.now();  
LocalDate maniana = hoy.plus(1, ChronoUnit.DAYS);  
LocalDate proximoPago = hoy.plus(2, ChronoUnit.MONTHS);
```



Mañana es hoy
más 1 día



El próximo pago
es dentro de 2
meses

Period

- Los periodos también se utilizan para calcular la diferencia entre dos fechas

```
LocalDate localDate1 = LocalDate.of(2019, Month.MAY, 18);  
LocalDate localDate2 = LocalDate.of(2019, Month.MAY, 20);  
Period period = Period.between(localDate1, localDate2);  
System.out.println(period.getDays());
```

Trabajo con Tiempos

- De forma similar a **LocalDate** se tiene **LocalTime**
 - Haciendo uso del método **of()**, esta clase puede crear un **LocalTime** teniendo en cuenta:
 - hora y minuto;
 - hora, minuto y segundo
 - hora, minuto, segundo y nanosegundo.

```
LocalTime time = LocalTime.of(5, 30, 45, 35); //05:30:45:35
System.out.println(time.getHour()); //5
System.out.println(time.getMinute()); //30
System.out.println(time.getSecond()); //45
System.out.println(time.getNano()); //35
```

Trabajo con Tiempos

- De la misma forma que con `LocalDate`, se puede consultar la hora actual

```
LocalTime time = LocalTime.now();
```

Duration

- Representa una diferencia de tiempo (similar a Period para fechas)

```
LocalTime localTime1 = LocalTime.of(12, 25);  
LocalTime localTime2 = LocalTime.of(17, 35);  
Duration duracion = Duration.between(localTime1, localTime2);  
System.out.println(duracion.getSeconds());
```

- También se puede crear un objeto Duration con el método of

```
Duration duracionDeUnDia = Duration.of(1, ChronoUnit.DAYS);  
Duration duracionDeUnDia_2 = Duration.ofDays(1);
```

Símbolo	Significado	Presentación	Ejemplos
a	Am-pm	Texto	PM
h	Hora 1-12	Número	12
K	Hora 0-11	Número	0
k	Hora 1-24	Número	15
H	Hora 0-23	Número	0
m	Minuto de la hora	Número	30
s	Segundo del minuto	Número	55
A	Millisegundo del día	Número	1234
V	Zona de tiempo	Zone-id	America/Los_Angeles; Z; -08:30
z	Zona de tiempo	Zone-name	Pacific Standard Time; PST
O	Zone-offset	offset0	GMT+8; GMT+08:00; UTC-08:00;

Fecha y Tiempo juntos

- La clase `LocalDateTime` es una clase compuesta que permite capturar un instante de tiempo completo

```
LocalDateTime dateTime = LocalDateTime.of(1989, 11, 11, 5, 30, 45, 35);
```

- También se puede crear un `LocalDateTime` a partir de un `LocalDate` y `LocalTime`

```
LocalDate date = LocalDate.of(1989, 11, 11);  
LocalTime time = LocalTime.of(5, 30, 45, 35);  
LocalDateTime dateTime = LocalDateTime.of(date, time);
```

- Finalmente puedo pedir el instante actual

```
LocalDateTime dateTime = LocalDateTime.now();
```